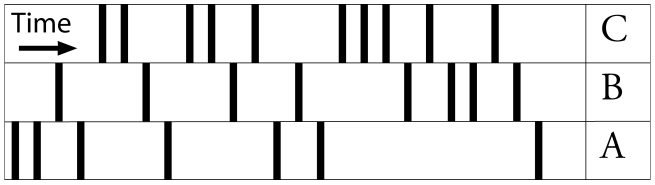
Modeling Player Performance In Rhythm Games

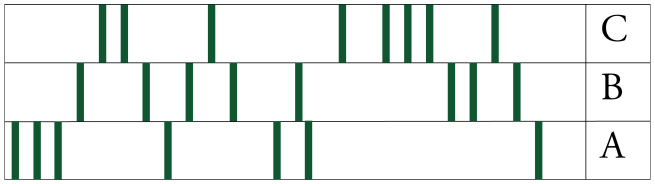
Lingfeng Yang*
Square-Enix Research Center

CR Categories: I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Modeling I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

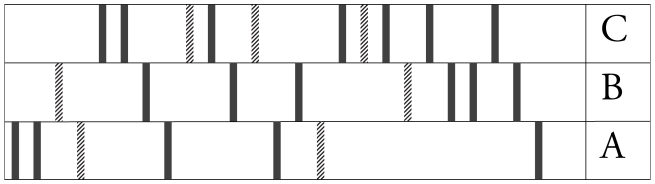
Keywords: interaction, games, music cognition, probabilistic inference



(a) Reference song



(b) User input



(c) Missed notes (shaded)

	A	B	C
A	0.17	0.33	0.5
B	0.5	0.125	0.375
C	0.11	0.55	0.33

	A	B	C
A	0	0.5	0.33
B	0.5	0	0.33
C	0.0	0.2	0.33

(d) (Left) Learned order-1 Markov transition table for $P(n_i|n_{i-1})$. (Right) Our corresponding skill model: miss rates $P(m_i|n_i, n_{i-1})$. Table layout: (row, column) $\rightarrow (n_{i-1}, n_i)$.

Figure 1: *The method at a glance: We represent songs with timelines, where A, B, C are the set of possible notes, and the bars represent specific occurences them in the song. (a) is a reference song and (b) is a performance of the song by the player. We combine (a) and (b) to obtain a labeling of missed notes (c) and learn a conditional probability distribution (d) describing the probability of missing a note given immediately previous notes.*

1 Introduction, Related Work

We present a method for modeling skill level in a rhythm game by building probabilistic models of how well the player is able to match a technically perfect execution. This enables applications for facilitating skill improvement in rhythm games.

*e-mail: lyang@cs.stanford.edu

SIGGRAPH Asia 2010, Seoul, South Korea, December 15 – 18, 2010.
ISBN 978-1-4503-0439-9/10/0012

Player modeling. Recently, game companies have shown interest in crafting games that adapt to individual players, driving academic research efforts in player modeling. Some work in this area focuses on modeling player satisfaction in platform games [Pedersen et al. 2009]. Other work is concerned with modeling player performance and using it to predict player behavior [Drachen et al. 2009]. Here, we are interested in modeling player performance in rhythm games and how the models can be used to facilitate skill development.

Algorithmic music and composition. A primary research problem in algorithmic music is the generation of complete musical pieces in a given style. This often involves building statistical models of music. A survey of modeling and generation techniques can be found in [Conklin 2003]. Here, we are interested in how statistical music models can be adapted for modeling music performance in the rhythm game setting. As a starting point, we adapt a Markov model. In future work, we hope to apply this approach with more complex models to realistic settings.

2 Modeling Rhythm Game Performance

2.1 Background

In a rhythm game, the player hits notes by pressing buttons on an input device. The goal is to hit notes in time to the music, which can be represented as a sequence S of note, time pairs:

$$S := \{s_i = (n_i, t_i), n_i \in C, t_i \in \mathbb{R}, t_1 < t_2 < \ldots t_N\}_{i=1}^{N},$$

where C is a finite set of possible notes, and N the total number of notes in the song. *Chords*, or sets of in-game notes that occur at the same t_i, are represented as a distinct note.

During the game, notes n_i in S appear on screen in chronological order. The player then receives visual and aural cues to hit each n_i at time t_i. The player is allowed to press the buttons corresponding to the notes in any way he wishes. After all the notes from S have appeared, the player will have produced a sequence of button presses

$$I := \{(n_i, t_i), n_i \in N, t_i \in \mathbb{R}\}_{i=1}^{M}$$

where M is the number of notes actually hit.

We consider a measure of skill to be a measure of how closely S and I match. Moreover, we are also concerned with *informative* critera; those that the player can improve intuitively and directly. Although there are many ways to describe the differences between S and I, the one we focus on is the *missed note*; where there is no $i_j \in I$ that matches a particular $s_k \in S$ in the time component (within a reasonable timing window).

For each $s_i \in S$, we compare with all $i \in I$ and label it as *missed* if there is no close enough i. This definition is useful because it covers many different kinds of mistakes, from playing the wrong note, to getting the rhythm of the song incorrect, but is still informative. A limitation of this labeling is that it does not capture the class of mistakes where the player plays too *many* notes. Nevertheless, we will see that tracking missed notes still gives useful information about the skill of a player and where it needs improvement.

2.2 Determining miss rate from a Markov model.

We first learn an order-k (we currently use $k = 1$) Markov chain that records the conditional probability of each note occuring in the reference song given the previous notes. The states of our Markov model are the notes $n_i \in C$; i.e., in this model, n_i (in $s_i = (n_i, t_i)$) is conditionally independent of all others given n_{i-1}.

As the player plays through S, he generates input I and missed notes of S are labeled. Let m_i be an indicator random variable tracking whether s_i is missed. We consider m_i to be conditionally dependent on n_i, n_{i-1}. This gives rise to the conditional distribution of missed notes $P(n_i|m_i, n_{i-1})$ along with the overall conditional probability of missing $P(m_i|n_{i-1})$. Then the *miss rate* of n_i in the note sequence n_i, n_{i-1} is obtained using Bayes' rule and the aforementioned probabilities:

$$P(m_i|n_i, n_{i-1}) = \frac{P(n_i|m_i, n_{i-1})}{P(n_i|n_{i-1})} P(m_i|n_{i-1})$$

Learning the miss rate. We employ a frequency-based estimator to learn the conditional probabilities given a song-input pair S, I. If the order of the Markov model k is sufficiently small, useful conditional probabilities can be learned in the context of a single playthrough of a single song. See Figure 1 for an example of how the model is learned from user input given a small but sufficiently covering sequence of notes.

Variable-order models. As the player's proficiency increases, it becomes harder to capture technical shortcomings in a meaningful way using a Markov model with fixed order k. This is because the errors made depend on an increasingly longer history of notes. If there is a particular note sequence of length $n > k$ that the player consistently makes mistakes on, but the model is order k, and the player does not usually make mistakes on the last k notes of that sequence, the miss rate for that problematic sequence will be artifically low. Conversely, calculating miss rates using order k for sequences where $m < k$ would do suffers from the sparse data problem. This suggests augmenting our technique with *variable-order Markov models* (also used extensively in music modeling), which allow for each state to be dependent on a varying number of previous states instead of a fixed k states, and for which learning algorithms that achieve information-theoretic optimums exist [Begleiter et al. 2004].

3 Applications

After learning these models, they can be used in variety of applications for facilitating rhythm game skill development.

3.1 Difficulty prediction.

By annotating songs with their inferred difficulty, the player can know in advance which songs and which parts of the song will be hard to play, and thus allocate practice time more efficiently. In particular, we can estimate the probability with which any small sequence of notes in an arbitrary song will be played without errors. Visualizations may also be built from this data, such as assigning each note in the song a color corresponding to its miss rate given the previous notes.

Songs may also be ranked according to difficulty. One way to do this is to, given a Markov model of played songs along with miss rates, make the rank equal to a utility function of the expected rate at which the player will miss notes during the song. The utility function would ensure that the top-ranked song is difficult, but not too difficult.

3.2 Exercise generation.

Musicians have traditionally turned to technical exercises to improve skill. As the musician gains proficiency, shortcomings in technique are identified by the musician or his instructor, and technical exercises are prescribed to address them.

It is often not clear to the rhythm game player which exercises should be played in order to improve the fastest. Moreover, it is also often the case that no existing exercise addresses the unique technical shortcoming of the player; one then has to put in manual work to devise individualized exercises. With our skill model, it is possible to automate the generation of such exercises. One way is to simply generate repetitions of the most-often-missed note sequences. More sophsticated exercise generation algorithms would draw on techniques from algorithmic music.

4 Prototype Game

As a platform for testing these ideas, we developed a rhythm game whose mechanics are based on that of the popular DJ simulator Beatmania IIDX. We chose Beatmania IIDX because it already has an established repretoire of songs, simple button-based controls, and an active mod/content creation community. In our prototype, the model's parameters are trained simply by having the user play the game normally. The advantage of this is that the training process requires no additional interaction beyond what the user already expects to do playing a rhythm game. After each song is played, the game records the statistics for that song and appends it to a database. An exercise is generated from the statistics after each song is played, and the player has the option of using it for practice.

5 Discussion, Future Work

An in-depth user study needs to be conducted to quantify the degree to which the exercise generation and difficulty prediction features are effective in improving skill and accuracy, and to compare different methods of generating the exercise material in order to select the best method.

One avenue for future work is to augment this approach with more complex models drawn from algorithmic music generation, such as stochastic L-system grammars, which can capture longer-ranged aspects of compositions (and potentially, technique). Another is to take the idea of skill modeling in general and apply it to other types of games.

An interesting long-term direction is to use this approach to assist training in actual musical instruments. Automating the creation of individualized exercises based on player performance could potentially result in much faster skill development for all musicians.

References

BEGLEITER, R., EL-YANIV, R., AND YONA, G. 2004. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research 22*, 1, 385–421.

CONKLIN, D. 2003. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, Citeseer, 30–35.

DRACHEN, A., CANOSSA, A., AND YANNAKAKIS, G. 2009. Player modeling using self-organization in tomb raider: underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG2009), Milano, Italy. http://www. itu. dk/˜ yannakakis/CIG09_IOI. pdf*.

PEDERSEN, C., TOGELIUS, J., AND YANNAKAKIS, G. 2009. Modeling player experience in super mario bros. In *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009*, 132–139.